The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

WalkRanker: A Unified Pairwise Ranking Model with Multiple Relations for Item Recommendation

Lu Yu,* Chuxu Zhang,+ Shichao Pei,* Guolei Sun,* Xiangliang Zhang*

*King Abdullah University of Science and Technology, Thuwal, 23955, SA
+University of Notre Dame, IN 46556, USA
Email: lu.yu,shichao.pei,guolei.sun,xiangliang.zhang@kaust.edu.sa, czhang11@nd.edu

Abstract

Top-*N* item recommendation techniques, e.g., pairwise models, learn the rank of users' preferred items through separating items into positive samples if user-item interactions exist, and negative samples otherwise. This separation results in an important issue: the extreme imbalance between positive and negative samples, because the number of items with user actions is much less than those without actions. The problem is even worse for "cold-start" users. In addition, existing learning models only consider the observed *user-item* proximity, while neglecting other useful relations, such as the unobserved but potentially helpful *user-item* relations, and high-order proximity in *user-user*, *item-item* relations. In this paper, we aim at incorporating multiple types of user-item relations into a unified pairwise ranking model towards approximately optimizing ranking metrics *mean average precision* (MAP), and *mean reciprocal rank* (MRR). Instead of taking statical separation of positive and negative sets, we employ a *random walk approach* to dynamically draw positive samples from short random walk sequences, and a rank-aware negative sampling method to draw negative samples for efficiently learning the proposed pairwise ranking model. The proposed method is compared with several state-of-the-art baselines on two large and sparse datasets. Experimental results show that our proposed model outperforms the other baselines with average 4% at different top-*N* metrics, in particular for cold-start users with 6% on average.

Introduction

Providing personalized services to users still raises continuous challenges. Usually users' preferences over items could be encoded as *explicit* ratings like starring items or in the case of *implicit* data of *user-item interactions* (e.g., "user A clicks on or purchases item B"). Nowadays it's widely acceptable to pay more attention on optimizing the *rank* of items from implicit data, other than rating estimation (Koren, Bell, and Volinsky 2009). This leads to a shift from rating estimation to rank optimization. Inspired by *learning to rank* community, pairwise models towards rank optimization are among the most popular top-*N* item recommendation techniques, under the basic assumption that items without user actions (e.g., clicks) are of less interest than those with user actions.

Most of the pairwise learning approaches (Shi et al. 2012b; Rendle et al. 2009; Yu et al. 2016b; Zhang et al. 2013) separate items into positive or negative set according to the first-order local structure, if representing observed user-item interactions as a collaborative bipartite graph. It means that only direct neighbors (*i.e., items*) connected to a user u are allocated to a positive set, while all other items (not connected to u) are negative samples. Several issues might be raised by such a separation.

1. The ratio of negative samples to positive samples is seriously imbalanced, as users normally only have actions on a small portion of items. Especially, for those *cold-start* users, too few positive samples could lead to insufficiently modeling their preferences over items.
2. Only observed *user-item* relations are taken into account to construct pairwise samples, while those potentially useful but unobserved relations are ignored. These ignored relations include i) unobserved but potentially helpful *user-item* relations among users and items that are not directly connected in the bipartite graph; and ii) high-order *user-user, item-item* relations. *User-user* similarity and *item-item* similarity are essential for inferring user preference in collaborative filtering. However, the current separation in pairwise models restrains the consideration of high-order proximity.

Our goal in this work is to take **multiple relations** including *user-item, user-user* and *item-item* relations, into the rank learning model for improving top-*N* item recommendation.

To extract and measure multiple relations, we propose to generalize the 0/1 user-item relations considered in standard pairwise models to probability-based user-item relations (called *probability relevance*). From a probability perspective, we enable unobserved user-item pairs to become potentially positive samples if they can be connected indirectly in a bipartite graph. The introduction of *probability relevance* drives us to revisit the definition of previous *MAP* and *MRR Loss* for recommendation tasks where only binary relevance is considered. To smooth the newly defined loss functions, we derive their lower bound and find that new MAP and MRR loss function have the same lower bound. Therefore, we present a unified ranking model as an alternative optimization objective. To efficiently learn model parameters, sequential samples induced from random walk sequences are constructed as positive samples, and a rank-

aware dynamic negative sampling approach is employed to select effective samples from massive possible candidates. As the proposed method combines the characteristics of random walk and ranking model, it is named WALKRANKER. The main contributions of this work include:

1. We unify *MAP* and *MRR Loss* in a general pairwise ranking model, and integrate multiple types of relations for better inferring user's preference over items.
2. As far as we know, this is the first work that derives pairwise samples for multiple relations via generalizing the 0/1 relations to probability-based relations through a random walk strategy in a user-item bipartite graph.
3. Evaluation results show that the proposed model outperforms other rank learning methods. Particularly, it performs well on *cold-start* users with average 6% improvement on different top-*N* metrics over strongest baseline.

Related Work

At early period of time, the mainstream of recommendation research has focused on the rating estimation based on the explicit data (Koren, Bell, and Volinsky 2009; Linden, Smith, and York 2003). Recently many works have began to adopt the idea of learning-to-rank to explore users' preference on items from ranking perspective (Shi, Larson, and Hanjalic 2010; Rendle et al. 2009; Hu, Koren, and Volinsky 2008; Zhang et al. 2017; Yu et al. 2016a). In contrast to rating prediction, addressing item recommendation as a ranking problem closely matches practical applications, but still remains many challenges. A typical one is how to deal with missing data. Hu *et al.* (Hu, Koren, and Volinsky 2008) proposed to weight unknown data with a prior value. Though the processed feedback matrix becomes very dense, only observed feedback matters after mathematical transformation. Also learning-ratio-free feature inspires a series of works such as (He et al. 2016; Bayer et al. 2017), which employ coordinate descent to reduce its computation complexity and obtain state-of-the-art performance. From the pairwise learning community, unequal relationship could be extracted between observed and unobserved feedback. BPR (Rendle et al. 2009) was a seminal research to model pairwise learning from Bayesian perspective. Later on, Rendle *et al.* (Rendle and Freudenthaler 2014) pointed out that the uniform sampling strategy implemented by BPR could suffer from insufficient gradient updates, and proposed an adaptive sampling method. Actually, before Rendle *et al.* (Rendle and Freudenthaler 2014), Zhang *et al.* (Zhang et al. 2013) discussed the sampling problem, and defined two types of dynamic sampling functions. Weston *et al.* (Weston, Bengio, and Usunier 2011) defined order pairwise ranking loss and developed online Weighted Approximate-Rank Pairwise (WARP) method, which can be applied to various top-*N* learning problems, such as video recommendation (Weston, Yee, and Weiss 2013), collaborative retrieval (Weston et al. 2012). Besides Bayesian pairwise preference defined in (Rendle et al. 2009), several pioneering works (Usunier, Buffoni, and Gallinari 2009; Shi et al. 2012b; 2012a; Weimer et al. 2007) attempt to directly take ranking metric favoring top-*N* performance as the optimization object. Shi *et al.* (Shi et al. 2012b; 2012a) proposed to directly optimize modified *MAP* and *MRR Loss*, but not utilizing rank positions of positive samples and keeping the binary relevance assumption that leads to static positive and negative sample set. RankMBPR (Yu et al. 2016b) is the most related work to WALKRANKER. However, WALKRANKER has several advantages over RankMBPR, e.g., it unifies both *MAP* and *MRR Loss* into a general pairwise ranking model, and proposes probability relevance to make it flexible to capture more types of relations to tackle data sparsity problem, especially for making cold-start recommendation.

Problem Formulation

Top-*N* item recommendation generates a list with *N* items that are ranked according to a user's preference (in descending order) inferred from users' historical feedbacks. Since no rating estimation is required, a variety of implicit feedback data can be used for learning the rank of top-*N* recommendations. Let U and I represent the user and item set, respectively. User-item interactions can be presented as a bipartite graph $G = (V, E)$, where $V = U \cup I$, and an edge $e_{ij} \in E$ only occurs between a vertex $v_i \in U$ and $v_j \in I$, where symbol i, j denotes the index for *i*th or *j*th vertex, respectively. Let $\mathcal{N}(v_i)$ denote the neighbors of vertex v_i. **Figure 1(a)** presents a toy example, where a circle vertex represents a user and a rectangle vertex denotes an item. Bipartite graph *G* can be represented as an adjacent matrix $\mathbf{X} \in \{0, 1\}^{|V|\times|V|}$, where each element $X_{ij} = 1$ if edge e_{ij} exists, otherwise $X_{ij} = 0$.

Unified Pairwise Ranking Model

In this section, we will first review the limitations of existing approaches, then elaborate the proposed method to overcome the challenges.

Motivation

Most of approaches for top-*N* recommendation task have two common features, i) Only observed user-item interaction is taken into account, while those unobserved but potentially helpful relations are ignored; ii) Positive and negative samples are statically separated, where observed feedbacks (*i.e.* $v_j \in \mathcal{N}(v_i)$) are regarded as positive samples, otherwise negative samples. In this work, we argue that the capacity of ranking approaches that follow these two features, may be limited based on the following consideration:

C1 Those ignored relations include i) *user-item* relation among users and items that are not directly connected in the bipartite graph. Taking user *u1* as an example, item *i2* and *i5* are both unvisited items, shown in **Figure 1(b)** and **1(c)**. Due to *i2*'s closeness to *u1*, *u1* may show a larger probability to have potential interaction with *i2* than *i5*; ii) though *user-user* and *item-item* links do not exist in the original graph, *user-user* similarity and *item-item* similarity can be inferred from their shared tastes on items or users. The inferred relations acting as regularization terms have been proved powerful in rating estimation tasks (Ma et al. 2011).

C2 Static division will cause imbalanced portion of positive to negative samples, because users usually have actions on a small portion of items. In particular, cold-start users have too few positive samples to sufficiently learn their preferences.

Model Multiple Relations (Response to C1)

To find the optimal rank order of items, a common approach is to optimize an objective function that is defined by a ranking metric. *Mean average precision (MAP)* is a widely-adopted measurement of the performance of a top-*N* recommendation method. We extend the original definition of *MAP* on *user-item* interaction (Shi et al. 2012a) to multiple relations (*i.e.,* user-item, user-user, item-item) as follows:

$$\begin{aligned} MAP &= \frac{\sum_{v_i \in V} AP_{v_i}}{|V|} \\ &= \sum_{v_i \in V} \frac{\sum_{v_j \in V} \frac{X_{ij}}{R_{ij}} \sum_{v_k \in V} X_{ik} \mathbb{I}(R_{ik} < R_{ij})}{|V| \sum_{v_j \in V} X_{ij}} \end{aligned} \tag{1}$$

where each vertex v_i can stand for a user or item, v_j acts as positive samples, and vertex v_k acts as a negative sample. R_{ij} denotes the rank of target v_j for a given entity v_i (e.g., $R_{ij} = 1$ if entity v_j ranks at the first), and $\mathbb{I}(\cdot)$ is the indicator function, which equals to 1 if the condition is satisfied, otherwise 0. Besides *MAP*, *mean reciprocal rank (MRR)* also aims at raising the most relevant document as top as possible in ranking. Similar to modified MAP loss, we modify the original definition (Shi et al. 2012b) for recommendation task as follows:

$$\begin{aligned} MRR &= \frac{\sum_{v_i \in V} RR_{v_i}}{|V|} \\ &= \sum_{v_i \in V} \sum_{v_j \in V} \frac{X_{ij}}{|V| R_{ij}} \prod_{v_k \in V} (1 - X_{ik} \mathbb{I}(R_{ik} < R_{ij})) \end{aligned} \tag{2}$$

Binary to Probability Relevance (Response to C2)

Note that only binary relevance $X_{ij} \in \{0, 1\}$ is considered when defining the above ranking metrics. Such an assumption makes optimization intractable, and fails to capture unobserved interactions as we discussed before. $X_{ij} = 0$ does not always mean that entity v_i has no proximity to v_j. It may be because v_i had no access to, but still probably be similar or like the unlinked entity v_j. Therefore, we propose to extend binary interaction with a probability relevance $\hat{X}_{ij} \in [0, 1]$ with the following details:

Definition 1 *For observed links (existing user-item interactions) in bipartite graph* G*, i.e., $\hat{X}_{ij} = 1$, which indicates that entity v_i always shows a clear preference over entity v_j. While for those unobserved links (user-item, user-user, item-item), instead of $\hat{X}_{ij} = 0$, the value of $\hat{X}_{ij}$ is a probability value between 0 and 1.*

In later section, we will present how to express $\hat{X}_{ij}$ based on a random walk method, with a basic idea that the value of $\hat{X}_{ij}$ has a positive relationship with the number of random paths starting from v_i towards v_j in bipartite graph G.

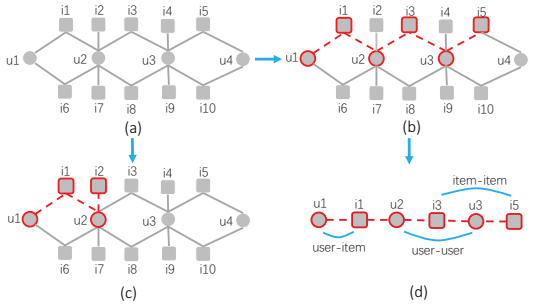

Figure 1: Toy example shows observed user-item relationship and latent interactions. (a) presents a bipartite graph derived from observed user-item interactions; (b) shows a path from source vertex *u1* to target item *i5*; (c) shows a path from source vertex *u1* to target item *i2*; (d) is an example path to show direct and indirect user-item, item-item, user-user interactions.

Expressing $\hat{X}_{ij}$ with dynamic random walk sequences will lead to dynamic division of positive and negative samples as a response to motivation **C2**.

Reciprocal Rank Optimization

Clearly, maximizing *MRR Loss* equals to optimizing each single component RR_{v_i}. Let's rewrite RR_{v_i} defined in Eq. (2) with probability relevance as follows:

$$\sum_{v_j \in V} \mathbb{E}_{(i,j) \sim p(i,j|\hat{X}_{ij})} \Big[\frac{1}{R_{ij}} \prod_{v_k \in V} (1 - \hat{X}_{ik} \mathbb{I}(R_{ik} < R_{ij})) \Big] \tag{3}$$

Measuring $\mathbb{I}(R_{ik} < R_{ij})$ in Eq. (3) actually equals to getting the mapping value of $\mathbb{I}(\hat{y}_{ik} > \hat{y}_{ij})$, where $\hat{y}_{ij}$ denotes the predicted relevance score for a pair of entities (v_i, v_j). To maximize RR_{v_i}, we need minimize $\hat{X}_{ik} \mathbb{I}(R_{ik} < R_{ij})$ as much as possible (i.e., $\hat{y}_{ik} < \hat{y}_{ij}$). Inspired by (Shi et al. 2012b), we can approximate $\hat{X}_{uj} \mathbb{I}(R_{uj} < R_{ui})$ by a sigmoid function $g(\hat{y}_{ikj})$, where $g(x) = 1/(1 + e^{-x})$, and $\hat{y}_{ikj} = \hat{y}_{ik} - \hat{y}_{ij}$. Analogously, we can approximate $1/R_{ij}$ with $g(\hat{y}_{ij})$, as a larger relevance score $\hat{y}_{ij}$ makes $1/R_{ij}$ closer to 1, while a smaller $\hat{y}_{ij}$ pushes $1/R_{ij}$ away from 1 (*e.g.*, ranking entity j at the bottom of the list). Note that the entities are ranked in descending order according to their predicted score. Then the smoothed RR_{v_i} optimizes recommendation model parameters Θ by:

$$\Theta := \arg\max_{\Theta} \{RR_{v_i}\} \Leftrightarrow \arg\max_{\Theta} \{\ln \frac{1}{|V|} RR_{v_i}\} \tag{4}$$

Applying Jensen's inequality on the concave function $\ln(\cdot)$, we derive the lower bound of $\ln(\frac{1}{|V|} RR_{v_i})$ as follows:

$$\begin{aligned} &\ln \Big(\frac{1}{|V|} \sum_{v_j \in V} \mathbb{E}_{(i,j) \sim p(i,j|\hat{X}_{ij})} \big[g(\hat{y}_{ij}) \prod_{v_k \in V} (1 - g(\hat{y}_{ikj})) \big] \Big) \\ &\geq \frac{1}{|V|} \sum_{v_j \in V} \mathbb{E}_{(i,j) \sim p(i,j|\hat{X}_{ij})} \big[\ln g(\hat{y}_{ij}) + \sum_{v_k \in V} \ln g(\hat{y}_{ijk}) \big] \end{aligned} \tag{5}$$

Finally we can approximately optimize *MRR* by maximizing following objective function:

$$\sum_{v_i \in V} \sum_{v_j \in V} \mathbb{E}_{(i,j)\sim p(i,j|\hat{X}_{ij})} \big[\ln g(\hat{y}_{ij}) + \sum_{v_k \in V} \ln g(\hat{y}_{ijk}) \big] \tag{6}$$

Average Precision Optimization

As shown in Eq. (1), *AP* is only based on the ranks of relevant items, but not including the irrelevant ones. According to the study in (Shi et al. 2012a), improving *AP* can be achieved via increasing the predicted value of relevant entities (user or item), meanwhile reducing the value of irrelevant ones. Therefore, maximizing the gap of ranks between relevant and irrelevant entities equals to ranking relevant ones before irrelevant as much as possible. Similar to the induction of approximated *MRR Loss* with employing smoothing function $g(\cdot)$, optimizing *MAP Loss* can be achieved by optimizing each component AP_{v_i} as below:

$$\Theta := \arg\max_{\Theta} \{AP_{v_i}\} = \arg\max_{\Theta} \{\ln \frac{1}{|V|^2} AP_{v_i}\} \tag{7}$$

Analogous to the induction of lower bound of RR_{v_i}, we derive the lower bound of $\ln \frac{1}{|V|^2} AP_{v_i}$ as follows:

$$\begin{aligned} &\ln \big(\frac{1}{|V|^2} \sum_{v_j \in V} \mathbb{E}_{(i,j)\sim p(i,j|\hat{X}_{ij})} \big[\frac{1}{R_{ij}} \sum_{v_k \in V} (1 - \hat{X}_{ik}) \mathbb{I}(R_{ij} > R_{ik}) \big] \big) \\ &\geq \sum_{v_j \in V} \mathbb{E}_{(i,j)\sim p(i,j|\hat{X}_{ij})} \big[\ln g(\hat{y}_{ij}) + \sum_{v_k \in V} \ln g(\hat{y}_{ijk}) \big] \end{aligned} \tag{8}$$

where we approximate the term $(1-\hat{X}_{ik})\mathbb{I}(R_{ik} < R_{ij})$ with a sigmoid function $g(\hat{y}_{ijk})$, and replace $\frac{1}{R_{ij}}$ with $g(\hat{y}_{ij})$.

Unified Pairwise Collaborative Filtering

Comparing the new *MAP Loss* optimization function in Eq. (8) with the final *MRR Loss* optimization function defined in Eq. (6), we see that they have the exact same formulation. This means that we can optimize two metrics in a unified objective model. Intuitively, $g(\hat{y}_{ijk})$ measures the probability that the predicted relevance value of positive sample v_j should be larger than negative one v_k. However, it has an issue with sufficiently penalizing the entities that are at a lower rank, which leads to a suboptimal top-*N* recommendation performance. A popular way to solve this problem is to employ a *weighting pairwise ranking loss* to give larger penalties on positive ones at lower rank (Yu et al. 2016b; Weston, Bengio, and Usunier 2011; Yuan et al. 2016). Therefore, we formulate the unified ranking objective function as follows:

$$\begin{aligned} \sum_{v_i \in V} \sum_{v_i \in V} &\mathbb{E}_{(i,j)\sim p(i,j|\hat{X}_{ij})} \big[\ln g(\hat{y}_{ij}) \\ &+ w_{ij} \sum_{v_k \in V} \ln g(\hat{y}_{ijk}) \big] - \lambda_\Theta ||\Theta||^2 \end{aligned} \tag{9}$$

where λ_Θ is the model regularization parameter, and w_{ui} denotes a *dynamic weight* depending on the rank of positive item *i* (described in later Section). Model parameters Θ can be learned via maximizing Eq. (9).

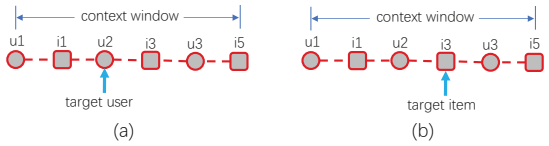

Figure 2: Illustration to describe a path generated via a random walk beginning at vertex *u1*, and possible interactions within a fixed context window.

Learning Algorithm

Although Eq. (9) can be solved by stochastic gradient methods, there remain two important challenges: (i) how to select a positive sample since multiple relations with probability relevance assumption taken into account; (ii) after selecting a positive sample, how to choose a negative one.

Optimize Ranking Model (Response to C2)

In this work, positive samples are derived from a random walk sequence. A pair of entities (e.g., *user-item, item-user, user-user, item-item*) are regarded as a positive training sample if they simultaneously appear in a random walk sequence, even they are not directly connected in the bipartite graph. Negative samples are dynamically sampled according to the rank position of positive samples. For each node $v_i \in V$, we can obtain a short random walk sequence $\mathcal{W}^\tau_{v_i} = \{w^0_{v_i}, w^1_{v_i}, w^2_{v_i}, \cdots, w^\tau_{v_i}\}$, $w^\tau_{v_i} \in V$ and $w^0_{v_i} = v_i$. After a sequence $\mathcal{W}^\tau_{v_i}$ is generated, training samples are extracted to learn model parameters. Given a random walk sequence $\mathcal{W}^\tau_{v_i}$, we can enumerate every $w^i_{v_i} \in \mathcal{W}^\tau_{v_i}$ as a target vertex, and uniformly generate a random window size ρ to construct a context $C^i_{v_i} = \{w^{i-\rho}_{v_i}, \cdots, w^{i-1}_{v_i}, w^{i+1}_{v_i}, \cdots, w^{i+\rho}_{v_i}\}$, shown as a toy example in **Figure 2**. Note that each vertex $w^i_{v_i} \in V$ can be either a user or an item. Suppose the target vertex $w^i_{v_i} \in U$ is a user vertex. A context vertex $c_{v_i} \in C^i_{v_i}$ can be picked out to capture *user-item* interactions if c_{v_i} is an item vertex, and to capture *user-user* interactions if c_{v_i} is a user vertex. Let d_c denote the distance from the target vertex $w^i_{v_i}$ to c_{v_i} in $C^i_{v_i}$. The larger the d_c is, the weaker interaction happens between the target vertex $w^i_{v_i}$ and the context vertex c_{v_i}. We define a weight function $\omega(d_c) = e^{\beta \cdot (1-d_c)}$ as a response. By using gradient ascent, the parameters are updated by:

$$\Theta \mathrel{+}= \alpha \cdot \omega(d_c) \cdot \big(\frac{\partial \ln g(\hat{y}_{ij})}{\partial \Theta} + w_{ij} \frac{\partial \ln g(\hat{y}_{ijk})}{\partial \Theta} - \lambda_\Theta \Theta \big) \tag{10}$$

where negative sample v_k is sampled from negative set $V^-_{v_i}$. For each sequence $\mathcal{W}^\tau_{v_i}$, $V^-_{v_i} = V \setminus \{\mathcal{N}(v_i) \cup \mathcal{W}^\tau_{v_i}\}$. Since $\mathcal{W}^\tau_{v_i}$ is dynamically generated, we can obtain a dynamic division of positive (*i.e.,* $\{\mathcal{N}(v_i) \cup \mathcal{W}^\tau_{v_i}\}$) and negative samples. Score function is $\hat{y}_{ij} = b_i + b_j + \mathbf{u}_i \mathbf{v}_j^T$, where $\mathbf{u}_* \in \mathbb{R}^d$, $\mathbf{v}_* \in \mathbb{R}^d$, and b_i is the bias term for each entity $v_i \in V$.

Weighted Dynamic Negative Sampling

To select effective negative samples for gradient updates, we employ a rank-aware negative sampling approach proposed in (Yu et al. 2016b). The most significant advantage of this approach is that the *dynamic weight* w_{ui} depends

Table 1: Statistics of the used datasets.

Datasets	#Users	#Items	#Feedbacks	Sparsity
Yelp	16,826	14,902	245,109	0.097%
Epinions	49,289	139,738	664,823	0.02%

on the rank position of positive samples, and is defined as $w_{ij} = \sum_{s=1}^{r_j} \frac{1}{s}$, where $r_j = \sum_{v_k \in V_{v_i}^-} \mathbb{I}[\ell + \hat{y}_{ik} \geq \hat{y}_{ij}]$, ℓ denotes a margin value. A larger r_i value implies a higher confidence to speed up the parameter update. It's inefficient to obtain the exact r_i according to the definition, and infeasible to be optimized with gradient methods. Instead, it can be fast estimated based on a geometry distribution. More specifically, we first select a subset $\hat{V}_{v_i}^- \subset V_{v_i}^-$ with a fixed size $n \ll |V_{v_i}^-|$. For a given sample (v_i, v_j), one uniformly draws a random negative sample from $\hat{V}_{v_i}^-$ until finding a v_k, which satisfies $\ell + \hat{y}_{ik} \geq \hat{y}_{ij}$. Then the r_i can be approximated as $r_i \approx \lfloor \frac{n-1}{K} \rfloor$, where $\lfloor \cdot \rfloor$ denotes the floor function and K is the number of steps to find item j. Correspondingly, an item buffer $buffer_{ij}$ with size κ stores every sampled negative item j. Finally, r_i can be approximated as $r_i \approx \lfloor \frac{n-1}{min(K,\kappa)} \rfloor$, and the final negative sample v_k to update model parameters will be selected from the top of the sorted $buffer_{ij}$ in descending order based on $\hat{y}_{ik}$.

Experimental Evaluation

In this section, we will describe our experimental setting, and present the experimental results, with comparison to different kinds of baseline methods.

Dataset & Baselines

To examine the capacity of our proposed method, we conduct experiments on two real-world datasets (Yelp[1] in Round 3 and Epinions[2]). Following recent published works (Rendle et al. 2009; He et al. 2017), we convert star rating into binary feedback by setting observed entries to 1, regardless of the specific rating values. We pre-filter users with at least 4 reviews. The statistical summarization of two datasets is described in Table 1. We mainly focus on top-*N* measures including Precision@*N* (Herlocker et al. 2004), Recall@*N* (Herlocker et al. 2004), MAP@*N* (Liang et al. 2016), MRR@*N* (Yu et al. 2013). Several types of state-of-the-art baselines are taken into consideration when comparing with WALKRANKER, including pointwise algorithm **WRMF** (Hu, Koren, and Volinsky 2008), pairwise ranking approaches **BPR** (Rendle et al. 2009), **AdaBPR** (Rendle and Freudenthaler 2014), **WARP** (Weston, Bengio, and Usunier 2011), **DNS** (Zhang et al. 2013), and **RankMBPR** (Yu et al. 2016b). We use Librec[3] (Guo et al. 2015) (a java library for recommender system) to run all of the algorithms, and implement the proposed approach WALKRANKER.

[1]https://www.yelp.com/dataset_challenge

[2]http://www.trustlet.org/downloaded_epinions.html

[3]http://www.librec.net/

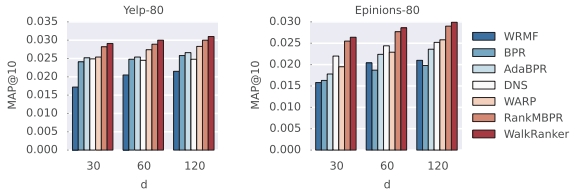

Figure 3: Recommendation performance on the full testing set with various values of dimension *d*.

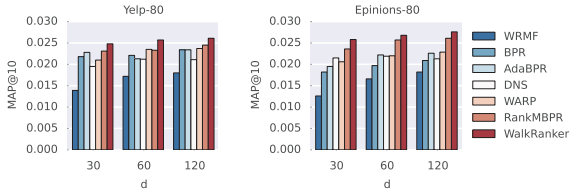

Figure 4: Recommendation performance on **cold-start users** with various values of dimension *d*.

Performance Analysis

To validate the performance of different algorithms on different data sparsity, we randomly select 80% or 50% data as training set, and the other 20% or 50% data as the testing set. All baseline algorithms in this work take matrix factorization as the scoring function. For the simplicity and fair comparison, we set the number of dimension parameter d to a same fixed number, *e.g.,* $d = 30$ for **Table 2** and **3**, $d = 60$ for **Figure 5** and **6**. In the "Data" column of **Table 2** and **3**, "dataname-80 or -50" like Yelp-80 denotes the proportion of raw data for training models. Though we preprocess Yelp users, there're still 49% users with less than 5 reviews in training set Yelp-80, and 68% in Yelp-50. All reported results are average value of 5 converged runs. The value of hyper-parameters for baselines are decided via implementing grid search on different settings, and the combination that leads to best performance is selected.

Adaptive Sampling. BPR is a strong pairwise baseline, and WRMF represents a state-of-the-art pointwise ranking algorithm. From the results shown in **Table 2** and **Figure 5**, we can see that usually BPR has a competitive performance against WRMF, but both of them are worse than those baselines with adaptive negative sampling approaches like AdaBPR, WARP and RankMBPR.

Rank-aware Strategy. Different from AdaBPR or DNS's dynamic sampling strategies, WARP, RankMBPR and WALKRANKER consider the rank of positive items as an indicator to control the parameters update magnitude. We can see that in both two datasets with different sparsity, algorithms with rank-aware strategy (e.g., RankMBPR and WALKRANKER) achieve better performance than AdaBPR.

With vs Without Multiple Relations. WALKRANKER incorporates more types of relations into a unified optimization function for top-*N* metrics. Comparing with

Table 2: Ranking performance comparison (the best results are marked with *). The last column shows the improvement of WALKRANKER over the best baseline algorithm, highlighted with underline.

Data	Metrics	WRMF	BPR	DNS	AdaBPR	WARP	RankMBPR	WALKRANKER	Improv.
Yelp-80	Precision@10	0.0153	0.0199	0.0203	0.0209	0.0221	0.0233	*0.0235	0.85%
	Recall@10	0.0427	0.0597	0.0607	0.0616	0.0625	0.0695	*0.0708	1.87%
	MAP@10	0.0172	0.0241	0.0249	0.0252	0.0254	0.0282	*0.0291	3.19%
	MRR@10	0.0507	0.0642	0.0663	0.0661	0.0675	0.0731	*0.0751	2.73%
Yelp-50	Precision@10	0.0232	0.0297	0.028	0.0305	0.031	0.0344	*0.0361	4.94%
	Recall@10	0.0331	0.0435	0.0395	0.0442	0.0469	0.0505	*0.0544	7.72%
	MAP@10	0.0165	0.0208	0.0204	0.0216	0.0218	0.0255	*0.0266	4.31%
	MRR@10	0.0459	0.0820	0.0806	0.0831	0.0855	0.0989	*0.1024	3.54%
Epinions-80	Precision@10	0.0168	0.0157	0.0208	0.0171	0.0191	0.0235	*0.0243	3.40%
	Recall@10	0.0359	0.0416	0.0504	0.0441	0.0477	0.0565	*0.0584	3.36%
	MAP@10	0.0158	0.0163	0.0220	0.0178	0.0195	0.0255	*0.0264	3.52%
	MRR@10	0.0538	0.0504	0.0684	0.0541	0.0581	0.0778	*0.0802	3.08%
Epinions-50	Precision@10	0.0256	0.0258	0.0320	0.0282	0.0284	0.0363	*0.0387	6.61%
	Recall@10	0.0278	0.0316	0.0361	0.0336	0.0350	0.0422	*0.0451	6.87%
	MAP@10	0.0162	0.0166	0.0211	0.0181	0.0185	0.0242	*0.0265	9.50%
	MRR@10	0.0772	0.0735	0.0909	0.0808	0.0788	0.1058	*0.1155	9.16%

Table 3: Detailed recommendation performance on cold-start users with no more than 5 feedbacks. Improvement of WALKRANKER over the best baseline is given in the last column.

Data	Metrics	WRMF	BPR	DNS	AdaBPR	WARP	RankMBPR	WALKRANKER	Improv.
Yelp-80	Precision@10	0.0062	0.0093	0.0088	0.0098	0.0093	0.0106	*0.0113	6.06%
	Recall@10	0.0389	0.0582	0.0544	0.0596	0.0578	0.0657	*0.0686	4.41%
	MAP@10	0.0139	0.0218	0.0195	0.0228	0.0210	0.0231	*0.0248	7.35%
	MRR@10	0.0219	0.0342	0.0313	0.0357	0.0323	0.0357	*0.0390	9.24%
Yelp-50	Precision@10	0.0098	0.0128	0.0113	0.0133	0.0139	0.0159	*0.0175	10.05%
	Recall@10	0.0328	0.0424	0.0368	0.0428	0.0453	0.0497	*0.0516	3.82%
	MAP@10	0.0122	0.0149	0.0138	0.0153	0.0164	0.0183	*0.0195	6.55%
	MRR@10	0.0369	0.0425	0.0387	0.0432	0.0476	0.0536	*0.0557	3.91%
Epinions-80	Precision@10	0.0052	0.0069	0.0078	0.0078	0.0081	0.0092	*0.0094	2.15%
	Recall@10	0.0353	0.0493	0.0518	0.0492	0.0537	0.0594	*0.0608	2.35%
	MAP@10	0.0126	0.0182	0.0215	0.0195	0.0206	0.0236	*0.0258	9.32%
	MRR@10	0.0194	0.0263	0.0318	0.0289	0.0293	0.0347	*0.0371	6.91%
Epinions-50	Precision@10	0.0085	0.0101	0.0112	0.0105	0.0113	0.0129	*0.0139	7.75%
	Recall@10	0.0272	0.0337	0.0366	0.0349	0.0361	0.0421	*0.0449	6.65%
	MAP@10	0.0107	0.0131	0.0144	0.0135	0.0138	0.0168	*0.0189	12.5%
	MRR@10	0.0304	0.0348	0.0381	0.0361	0.0385	0.0463	*0.0521	12.5%

RankMBPR, drawing pairwise samples from random walk sequence will bring several benefits, for example, the concept of context window extends the first-order to high-order proximity, and dynamic negative set adds flexibility to pairwise sample construction. The experimental results shown in **Table 2** demonstrate the effectiveness of the usage of multiple high-order relations. On average, WALKRANKER outperforms the strongest baseline RankMBPR 4% on different top-*N* metrics. From the results generated with 50% training set we can see that the proposed method has significant improvements over all baselines. Detailed evaluation with different lengths of recommendation list are in **Figure 5**, from which we can see that WALKRANKER is superior to different baselines on top-*N* metrics. **Figure 3** also demonstrates the effectiveness of WALKRANKER with different value of dimension *d*.

Cold-start Users. We further explore the performance of different algorithms on cold-start users with no more than 5 actions over items. As we said before, these users could have a serious imbalanced proportion between positive and negative samples. From the results shown in **Table 3**, we can see that the proposed method outperforms the best baseline with 6% on average at different measurements. In particular, the performance gap increases significantly with the decrease of training samples. More evidences can be found in **Figure 6** and **4**, from which we can find that WALKRANKER overall outperforms different baselines, especially brings significant improvement of recommendation quality for cold-start users on MAP@N and MRR@N metrics.

Parameter Sensitive Analysis

There are several hyper-parameters influencing the performance of the proposed method WALKRANKER. When validating the impact of a single parameter, we fix the others.

A. How long a random walk should take?

From the results shown in **Figure 7(a)**, we can see that the walk length has overall limited impact on the performance of WALKRANKER. The result slightly fluctuates with respect to different length limit on Yelp dataset. While, the performance of WALKRANKER appears to increase as the growth of walk length τ on Epinions data. It implies that a long walk sequence should be taken for a very sparse data like Epinions with 0.02% sparsity.

B. Large or small context window?

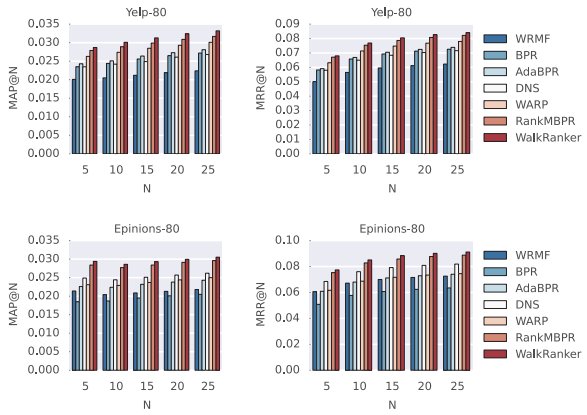

Figure 5: Top-N recommendation evaluation with different values of N.

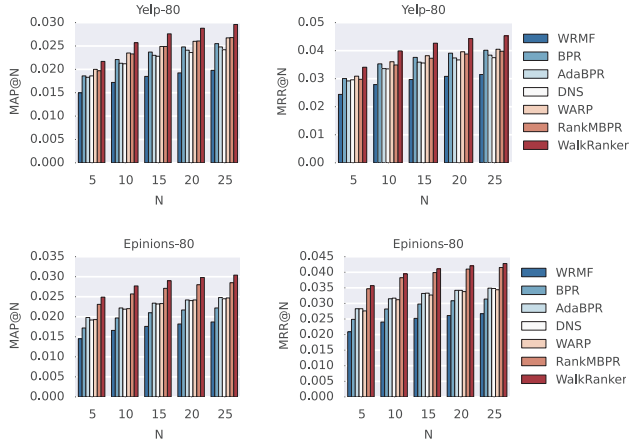

Figure 6: Top-N recommendation evaluation for **cold-start users** with different values of N.

For a given random walk sequence, we will enumerate each vertex as well as a context window defined to control the scope of local interactions. It works together with walk length τ to induce multiple relations. The results shown in **Figure 7(b)** indicate that $\tau = 10$ could be a suitable choice. Intuitively, a large context window might introduce noisy vertexes from long distance, and a small context window may lack the ability to incorporate high-order proximity.

C. Equally or unequally deal with context vertex?

To present the significance of a context vertex, we need to answer a question whether a context vertex should be equally or unequally regarded. In this work, we define a weight parameter β to control the decrease ratio with respect to a distance value. When $\beta = 0$, each context vertex has the same significance to the target vertex. The experimental results in **Figure 7(c)** demonstrate that context vertexes should not be considered equally. An empirical setting $\beta = 0.4$ leads to optimal performance.

D. How to control rank-aware sampling procedure?

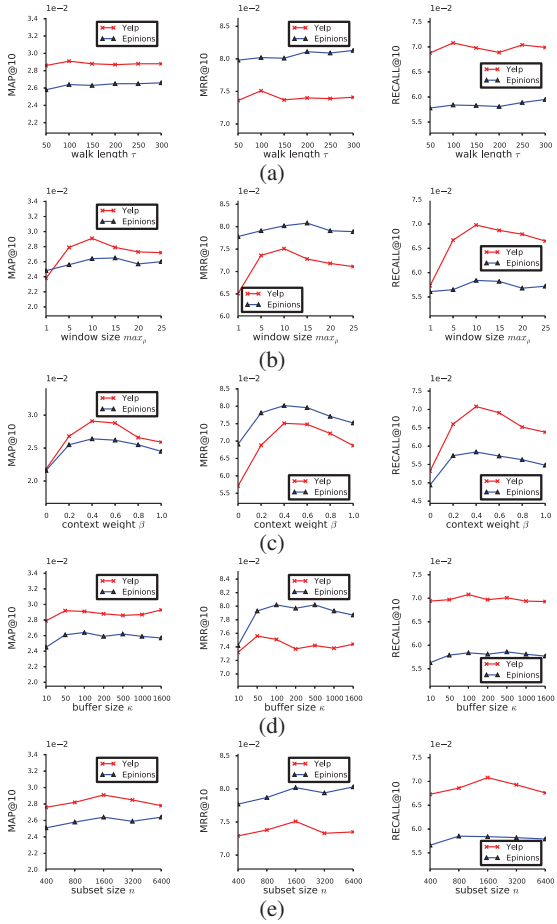

Figure 7: Impacts of different parameter settings on the performance of WALKRANKER.

Negative samples are uniformly selected in random from a subset with size n, and stored in a buffer with size κ. The rank of a positive sample could be estimated as $\lfloor \frac{n-1}{min(K,\kappa)} \rfloor$. We can see that the setting of parameters n and κ can decide the minimum value of the *dynamic weight* w. We should simultaneously study the effects of these two parameters on item recommendation performances. **Figure 7(d)** illustrates the performance variation over κ when fixing the subset size $n = 1600$. The performances of WALKRANKER tend to be stable when κ increases towards the value of $n = 1600$. **Figure 7(e)** shows that the performances of WALKRANKER vary with the subset size n. The main reason is that a large value of n could output a large weight to update parameters, which indirectly increases the variance of gradients. However, if we have a database with large scale amount of items like Epinions, we could safely try a large subset size n as the results shown in **Figure 7(e)**. In this work, we set $\kappa = 100$, subset size $n = 1600$ for Yelp, and $n = 6400$ for Epinions.

Conclusions

In this paper, we aim at incorporating multiple types of user-item relationships into a unified pairwise ranking model to-

wards approximately optimizing MAP and MRR ranking metrics. To tackle the challenges raised from the static division of pairwise samples, we propose to represent multiple types of relationship from a probability perspective. Under this assumption, we modify the discrete definition of *MAP* and *MRR Loss*, then present a unified pairwise ranking model for optimizing top-*N* recommendation performance. We also propose a simple, efficient way to construct training samples to reflect the probability relevance from a random walk perspective. Multiple relations induced from short random walk sequences tackle the challenges coming from data sparsity problem, in particular cold-start users. We experimentally demonstrated the importance of the combination of dynamic sampling strategy and multiple relations to solve the problem of item recommendation from implicit feedback. In future, we would like to explore the potential application of our method on entity recommendation and relation learning in a heterogeneous network, where simple random walk will be biased in training sample construction.

Acknowledgement

This work is supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. 2639.

References

Bayer, I.; He, X.; Kanagal, B.; and Rendle, S. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW'17*, 1341–1350.

Guo, G.; Zhang, J.; Sun, Z.; and Yorke-Smith, N. 2015. Librec: A java library for recommender systems. In *UMAP'15 Workshops*.

He, X.; Zhang, H.; Kan, M.-Y.; and Chua, T.-S. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR'16*, 549–558.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW'17*, 173–182.

Herlocker, J. L.; Konstan, J. A.; Terveen, L. G.; and Riedl, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22(1):5–53.

Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM'08*, 263–272.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer* 42(8):30–37.

Liang, D.; Altosaar, J.; Charlin, L.; and Blei, D. M. 2016. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys'16*, 59–66.

Linden, G.; Smith, B.; and York, J. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7(1):76–80.

Ma, H.; Zhou, D.; Liu, C.; Lyu, M. R.; and King, I. 2011. Recommender systems with social regularization. In *WSDM'11*, 287–296.

Rendle, S., and Freudenthaler, C. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM'14*, 273–282.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI'09*, 452–461.

Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Hanjalic, A.; and Oliver, N. 2012a. TFMAP: Optimizing map for top-n context-aware recommendation. In *SIGIR'12*, 155–164.

Shi, Y.; Karatzoglou, A.; Baltrunas, L.; Larson, M.; Oliver, N.; and Hanjalic, A. 2012b. CLiMF: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys'12*, 139–146.

Shi, Y.; Larson, M.; and Hanjalic, A. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *RecSys'10*, 269–272.

Usunier, N.; Buffoni, D.; and Gallinari, P. 2009. Ranking with ordered weighted pairwise classification. In *ICML'09*, 1057–1064.

Weimer, M.; Karatzoglou, A.; Le, Q. V.; and Smola, A. 2007. COFIRANK maximum margin matrix factorization for collaborative ranking. In *NIPS'07*, 1593–1600.

Weston, J.; Bengio, S.; and Usunier, N. 2011. WSABIE: Scaling up to large vocabulary image annotation. In *IJCAI'11*, 2764–2770.

Weston, J.; Wang, C.; Weiss, R.; and Berenzweig, A. 2012. Latent collaborative retrieval. In *ICML'12*, 9–16.

Weston, J.; Yee, H.; and Weiss, R. J. 2013. Learning to rank recommendations with the k-order statistic loss. In *RecSys'13*, 245–248. ACM.

Yu, X.; Ren, X.; Sun, Y.; Sturt, B.; Khandelwal, U.; Gu, Q.; Norick, B.; and Han, J. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *RecSys'13*, 347–350.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016a. A dynamic recurrent model for next basket recommendation. In *SIGIR'16*, 729–732.

Yu, L.; Zhou, G.; Zhang, C.; Huang, J.; Liu, C.; and Zhang, Z.-K. 2016b. RankMBPR: Rank-aware mutual bayesian personalized ranking for item recommendation. In *WAIM'16 (Best student paper award)*, 244–256.

Yuan, F.; Guo, G.; Jose, J. M.; Chen, L.; Yu, H.; and Zhang, W. 2016. LambdaFM: Learning optimal ranking with factorization machines using lambda surrogates. In *CIKM'16*, 227–236.

Zhang, W.; Chen, T.; Wang, J.; and Yu, Y. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *SIGIR'13*, 785–788.

Zhang, C.; Yu, L.; Wang, Y.; Shah, C.; and Zhang, X. 2017. Collaborative user network embedding for social recommender systems. In *SDM'17*.